Lab 4

COMP9021, Session 1, 2016

1 ☞ A triangle of characters

Write a program that gets a strictly positive integer N as input and outputs a triangle of height N, following this kind of interaction:

```
Enter strictly positive number: 13
            A
           BCB
          DEFED
         GHIJIHG
        KLMNONMLK
       PQRSTUTSRQP
      VWXYZABAZYXWV
     CDEFGHIJIHGFEDC
    KLMNOPQRSRQPONMLK
   TUVWXYZABCBAZYXWVUT
  DEFGHIJKLMNMLKJIHGFED
 OPQRSTUVWXYZYXWVUTSRQPO
ABCDEFGHIJKLMLKJIHGFEDCBA
```

Two built-in functions are useful for this exercise:

- `ord()` returns the integer that encodes the character provided as argument;
- `chr()` returns the character encoded by the integer provided as argument.

For instance:

```
>>> ord('A')
65
>>> chr(65)
'A'
```

Consecutive uppercase letters are encoded by consecutive integers. For instance:

```
>>> ord('A'), ord('B'), ord('C')
(65, 66, 67)
```

2 ☞ Pascal triangle

Write a program that prompts the user for a number N and prints out the first $N + 1$ lines of Pascal triangle, making sure the numbers are nicely aligned, following this kind of interaction.

```
$ python question_2.py
Enter a nonnegative integer: 3
   1
  1 1
 1 2 1
1 3 3 1
$ python question_2.py
Enter a nonnegative integer: 7
               1
             1   1
           1   2   1
         1   3   3   1
       1   4   6   4   1
     1   5  10  10   5   1
   1   6  15  20  15   6   1
 1   7  21  35  35  21   7   1
$ python question_2.py
Enter a nonnegative integer: 11
                                    1
                                 1     1
                              1     2     1
                           1     3     3     1
                        1     4     6     4     1
                     1     5    10    10     5     1
                  1     6    15    20    15     6     1
               1     7    21    35    35    21     7     1
            1     8    28    56    70    56    28     8     1
         1     9    36    84   126   126    84    36     9     1
      1    10    45   120   210   252   210   120    45    10     1
   1    11    55   165   330   462   462   330   165    55    11     1
```

3 ☞ Computing statistics on the characters in a text

Write a program that prompts the user for the name of a file and outputs how many times each digit occurs in this file, provided it does occur, following this kind of interaction:

```
$ cat test_1.txt
The Kiwis were the tournaments gallants, but this day were
overwhelmed, perhaps by the occasion,
certainly by Australia's brand of cricket forte.
Plan B ? sans McCullum's salvo ? had worked against
other attacks, but not the Australians' lair of limber lefties.

Beforehand, speculation centred on how the Kiwis,
playing away from their compact homelands
for the first time in the tournament, would deal
with the vastness of the MCG. Now, though, the problem
was not that the boundaries were too far away,
but the bowlers too close. Starc, and after him Mitch Johnson,
and must have looked like fishtailing trucks
coming towards them, with Josh Hazlewood swerving
from the other direction in the next lane.
"Our bowlers won us the World Cup," Clarke would aver later.
After six weeks of batting hit-and-giggle,
bowlers had the last laugh. They always do.

$ python question_3.py
Enter the name of a file: test_1.txt
There is no digit in this file.
$ cat test_2.txt
Chevron's decision to sell its 50 per cent stake in
Caltex Australia will make it easier for the local fuel
supplier to release franking credits to shareholders,
Caltex chief financial officer Simon Hepworth says.

Speaking after the $4.6 billion block sale, Caltex management
sought to assure investors that the company's
broader business strategy would be unchanged, despite the
departure of its US-domiciled major shareholder,
which has held its stake for 40 years.

But Mr Hepworth conceded the deployment of the company's
$1.1 billion franking credit balance could be
made easier by the transaction, given that as a US-based
shareholder, the return of franking credits was
not available to Chevron.
$ python question_3.py
Enter the name of a file: test_2.txt
Digits:    0   1   4   5   6
Count:     2   2   2   1   1
```

4 The Gale Shapley algorithm

Read the AMS Feature column on the stable marriage problem and the Gale Shapley algorithm.

Write a program that

- lets the user input the number n of couples,
- either lets the user input names or uses the default names M_1, ..., M_n for men and W_1, ..., W_n for women,
- either lets the user define preferences or randomly generates preferences.

If the preferences have been randomly generated then they are output. Finally, the Gale Shapley algorithm is applied and the matches are displayed.

Here is a possible interaction (the matches are given with women in first position, and lexicographically ordered):

```
>>> run_algorithm()
Enter a strictly positive number for the number of couples: 4

Enter 4 names for the men, all on one line and separated by spaces,
  or just press Enter for the default "names" M_1, ..., M_4:

Enter 4 names for the women, all on one line and separated by spaces,
  or just press Enter for the default "names" W_1, ..., W_4:

Press Enter to get a default preference for all men or women.
Otherwise, input one or more nonspace characters before Enter
  to be prompted and enter the preferences of your choice:

Preferences for M_1: W_3 W_1 W_4 W_2
Preferences for M_2: W_2 W_1 W_4 W_3
Preferences for M_3: W_2 W_4 W_1 W_3
Preferences for M_4: W_3 W_1 W_2 W_4

Preferences for W_1: M_2 M_3 M_4 M_1
Preferences for W_2: M_4 M_2 M_3 M_1
Preferences for W_3: M_1 M_4 M_3 M_2
Preferences for W_4: M_4 M_1 M_2 M_3

The matches are:
W_1 -- M_4
W_2 -- M_2
W_3 -- M_1
W_4 -- M_3
```

```
>>> run_algorithm()
Enter a strictly positive number for the number of couples: 4

Enter 4 names for the men, all on one line and separated by spaces,
  or just press Enter for the default "names" M_1, ..., M_4:

Enter 4 names for the women, all on one line and separated by spaces,
  or just press Enter for the default "names" W_1, ..., W_4:

Press Enter to get a default preference for all men or women.
Otherwise, input one or more nonspace characters before Enter
  to be prompted and enter the preferences of your choice: add

List preferences for M_1, in decreasing order: W_1 W_2 W_3 W_4
List preferences for M_2, in decreasing order: W_1 W_4 W_3 W_2
List preferences for M_3, in decreasing order: W_2 W_1 W_3 W_4
List preferences for M_4, in decreasing order: W_4 W_2 W_3 W_1

List preferences for W_1, in decreasing order: M_4 M_3 M_1 M_2
List preferences for W_2, in decreasing order: M_2 M_4 M_1 M_3
List preferences for W_3, in decreasing order: M_4 M_1 M_2 M_3
List preferences for W_4, in decreasing order: M_3 M_2 M_1 M_4

The matches are:
W_1 -- M_3
W_2 -- M_4
W_3 -- M_1
W_4 -- M_2
```

```
>>> run_algorithm()
Enter a strictly positive number for the number of couples: 2

Enter 2 names for the men, all on one line and separated by spaces,
  or just press Enter for the default "names" M_1, ..., M_2: paul john

Enter 2 names for the women, all on one line and separated by spaces,
  or just press Enter for the default "names" W_1, ..., W_2: kim lucy

Press Enter to get a default preference for all men or women.
Otherwise, input one or more nonspace characters before Enter
  to be prompted and enter the preferences of your choice: y

List preferences for paul, in decreasing order: kim lucy
List preferences for john, in decreasing order: lucy kim

List preferences for kim, in decreasing order: paul john
List preferences for lucy, in decreasing order: paul john

The matches are:
kim -- paul
lucy -- john
```